From a Virtualized Computing Nucleus to a Cloud Computing Universe: Data Management in the Cloud

Divy Agrawal
Department of Computer Science
UC Santa Barbara
http://www.cs.ucsb.edu/~agrawal
http://www.cs.ucsb.edu/~dsl

Collaborators: Amr El Abbadi, Sudipto Das, Aaron Elmore

A Voice from the Above

…Cloud Computing? What are you talking about? Cloud Computing is nothing but a computer attached to a network.

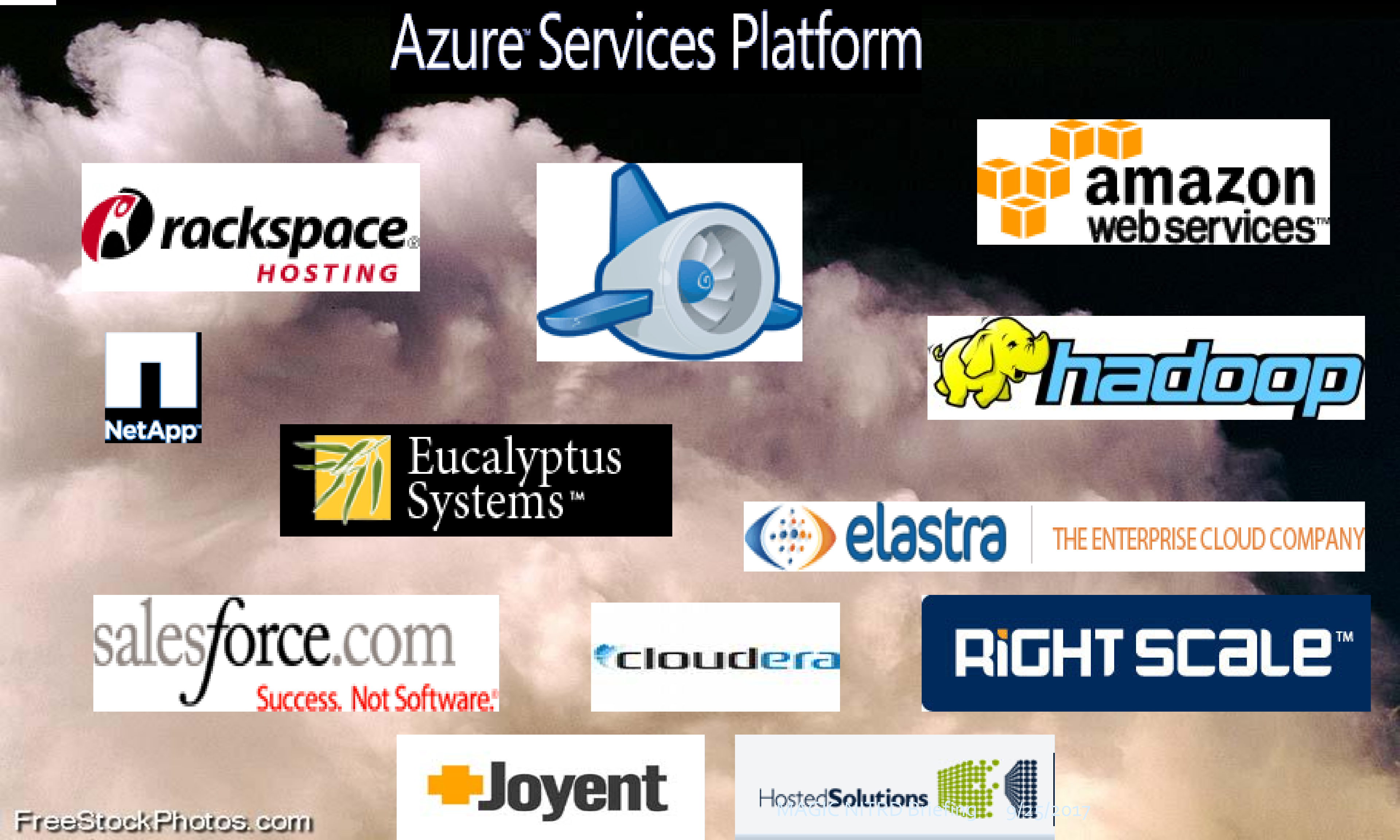
-- Larry Ellison, Excerpts from an interview

Outline

- Infrastructure Disruption
  - Enterprise owned → Commodity shared infrastructures
  - Disruptive transformations: Software and Service Infrastructure
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Transactional systems (Application Development)
  - Decision support system (Data Analysis)
- Cloudy Application Landscape
- Gen-next Data Management (UCSB)
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Virtualized Nucleus → Cloud Computing Universe

WEB is replacing the Desktop

Paradigm Shift in Computing

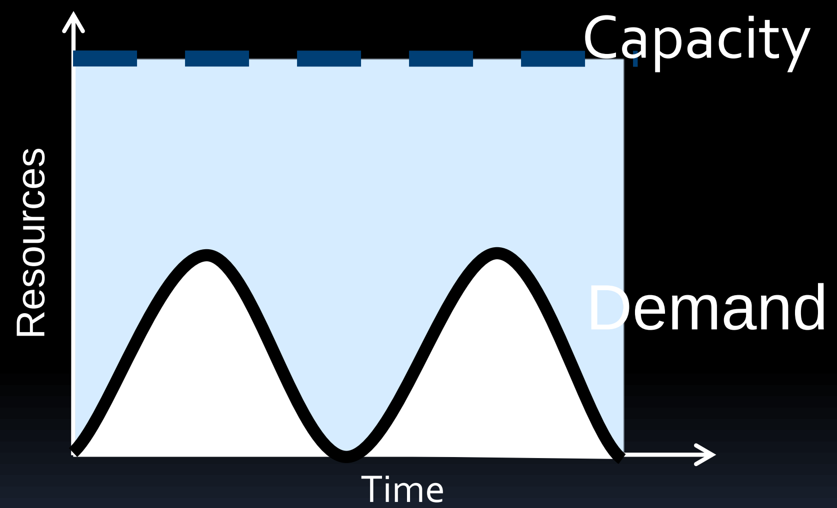


Cloud Computing: Why Now?

- Experience with very large datacenters
  - Unprecedented economies of scale
  - Transfer of risk
- Technology factors
  - Pervasive broadband Internet
  - Maturity in Virtualization Technology
- Business factors
  - Minimal capital expenditure
  - Pay-as-you-go billing model

Economics of Data Centers

- Risk of over-provisioning: underutilization



Static data center

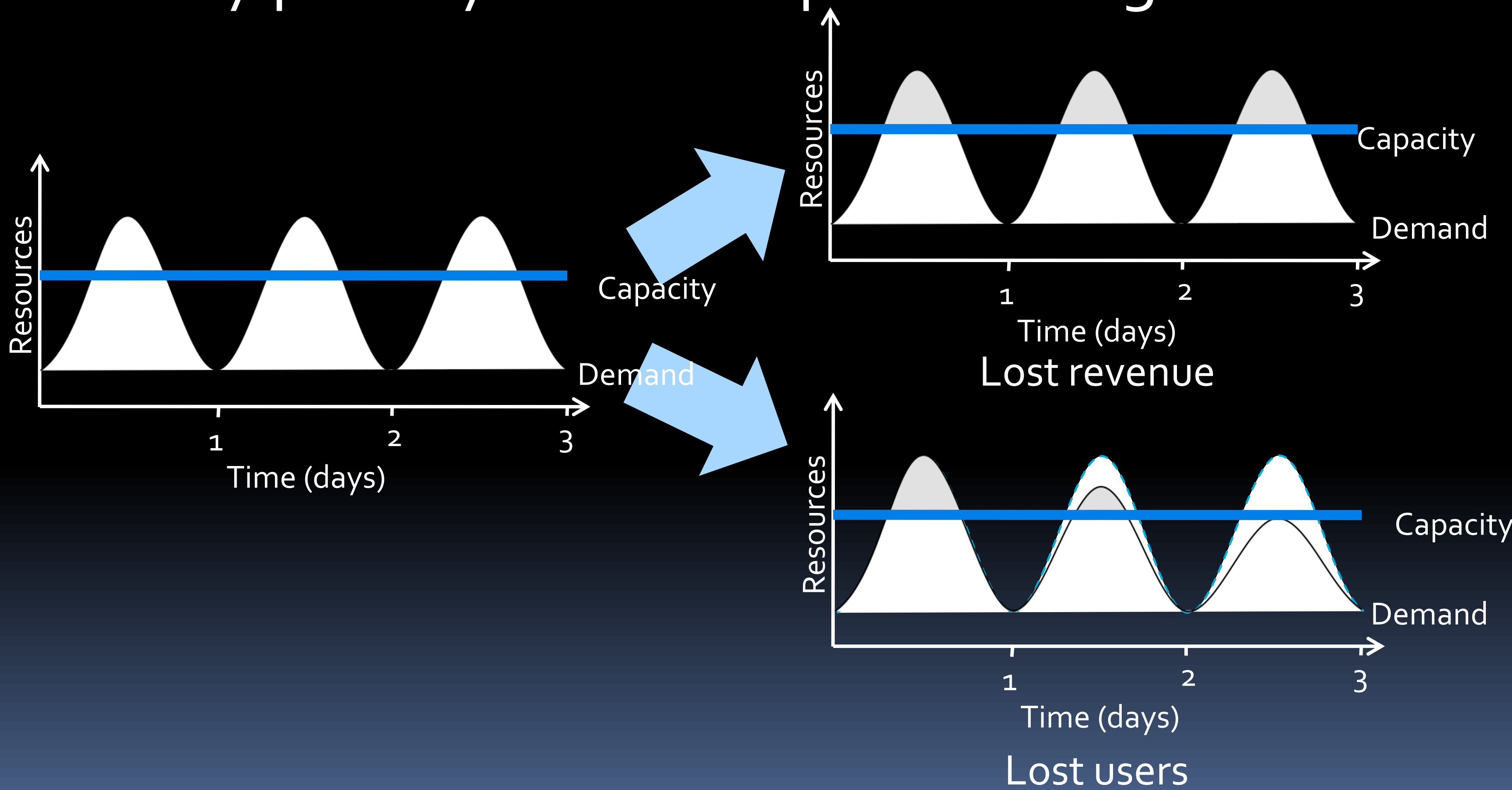
Money & Time Questions:

1. How much?

2. How Long?

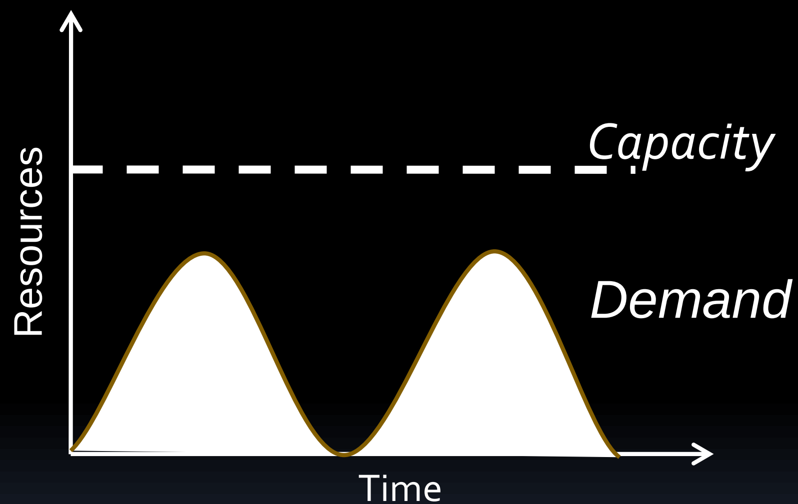
Economics of Internet Users

- Heavy penalty for under-provisioning



Economics of Cloud Computing

- Pay by use instead of provisioning for peak

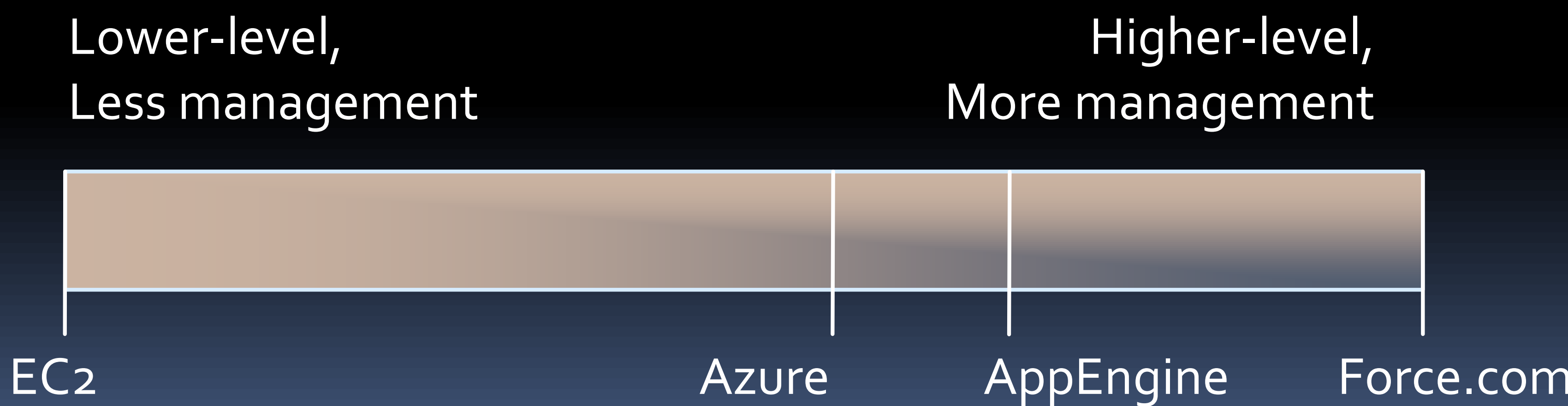


Static data center

Data center in the cloud

Cloud Computing Spectrum

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)



The Big Picture

- Unlike the earlier attempts:
  - Distributed Computing, Distributed Databases, Grid Computing
- Cloud Computing is REAL:
  - Organic growth: Google, Yahoo, Microsoft, and Amazon
  - IT Infrastructure Automation
  - Economies-of-scale
  - Fault-tolerance: automatically deal with failures
  - Time-to-market: no upfront invesment

Cloud Reality

- Facebook Generation of Application Developers
- Animoto.com:
  - Started with 50 servers on Amazon EC2
  - Growth of 25,000 users/hour
  - Needed to scale to 3,500 servers in 2 days (RightScale@SantaBarbara)
- Many similar stories:
  - RightScale
  - Joyent
  - …

Outline

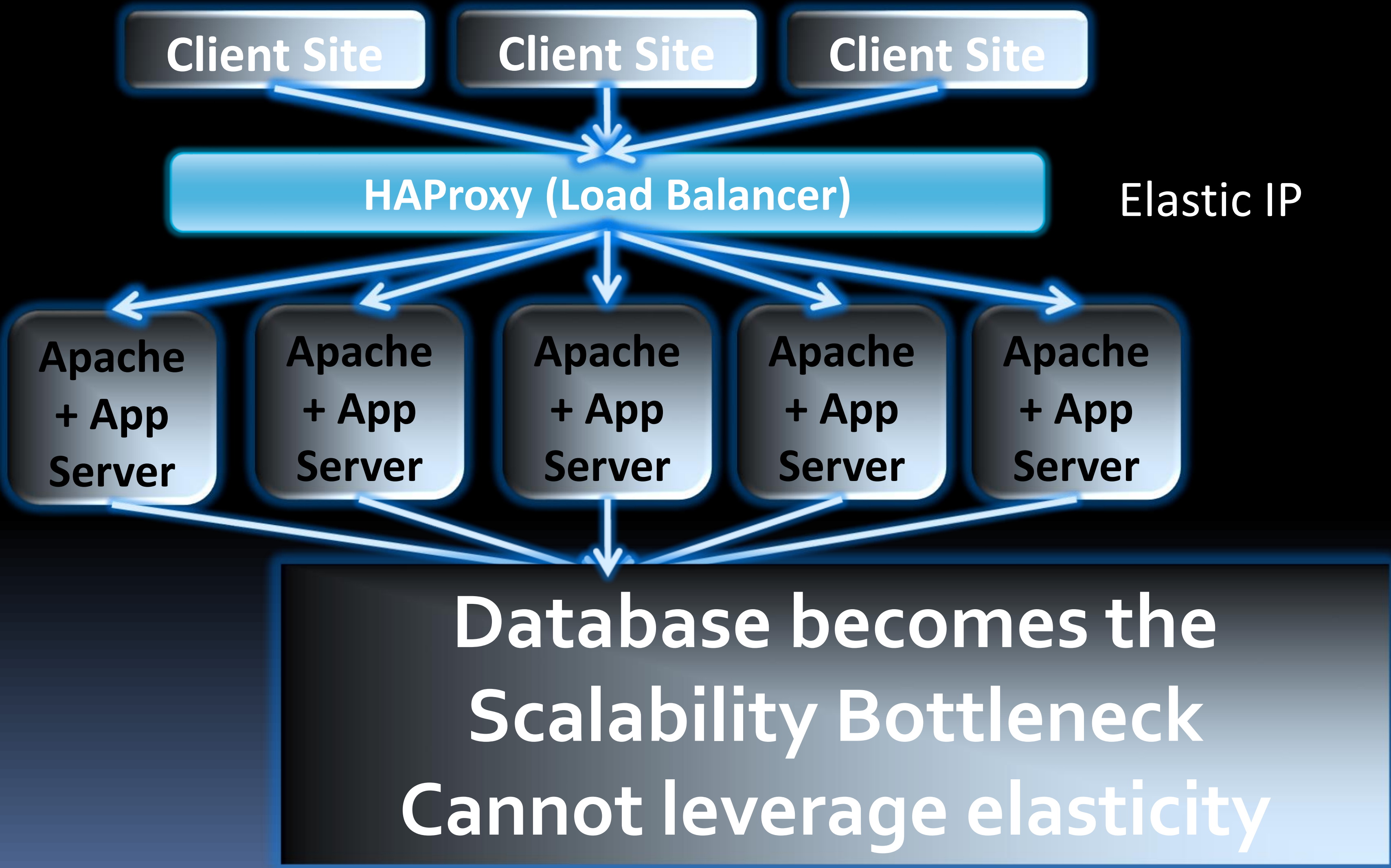
- Infrastructure Disruption
  - Enterprise owned → Commodity shared infrastructures
  - Disruptive transformations
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Transactional systems
  - Decision support system
- Cloudy Application Landscape
- Gen-next Data Management systems
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Virtualized Nucleus → Cloud Computing Universe

Current State

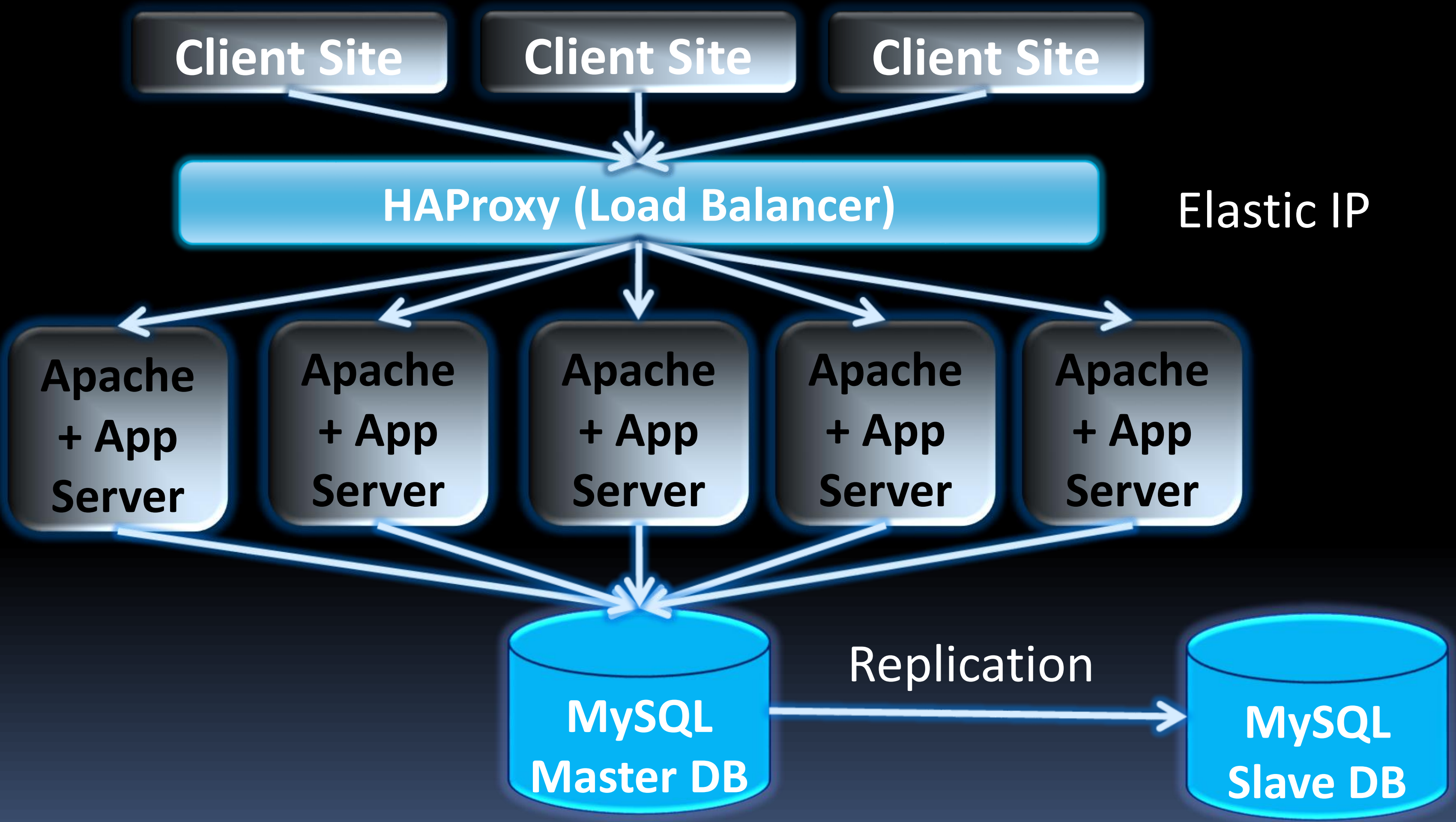
- Most enterprise solutions are based on RDBMS technology.
- Significant Operational Challenges:
  - Provisioning for Peak Demand
  - Resource under-utilization
  - Capacity planning: too many variables
  - Storage management: a massive challenge
  - System upgrades: extremely time-consuming
  - Complex mine-field of software and hardware licensing

➔ Unproductive use of people-resources from a company’s perspective

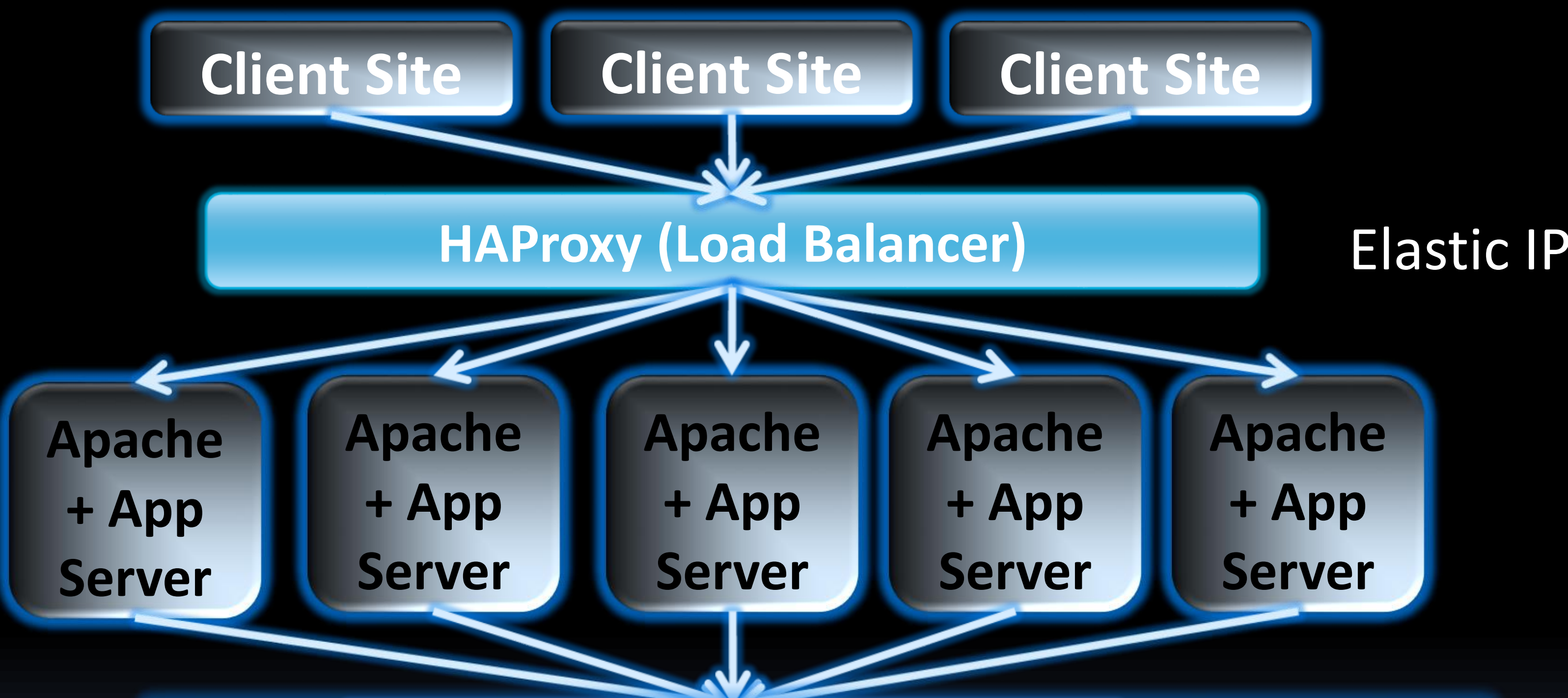
Scaling in the Cloud



Scaling in the Cloud



Scaling in the Cloud



Scalable and Elastic
But limited consistency and operational flexibility

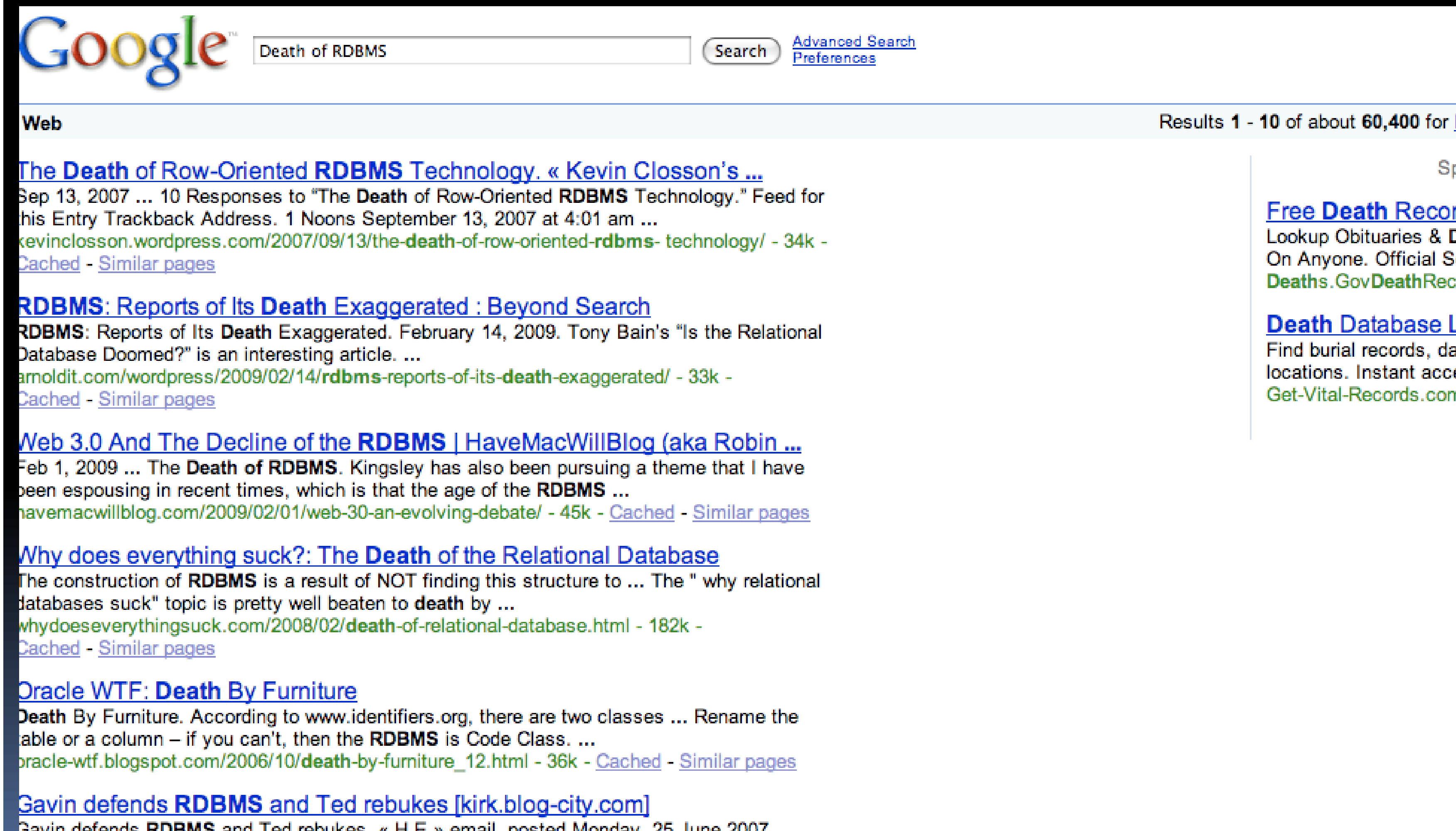
Cloud Computing Desiderata

- Scalability
- Elasticity
- Fault tolerance
- Self Manageability
- Sacrifice consistency?
  - Foregone Conclusion!!!

Outline

- Infrastructure Disruption
  - Enterprise owned => Commodity shared infrastructures
  - Disruptive transformations
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Transactional systems
  - Decision support system
- **Cloudy Application Landscape**
- Gen-next Data Management (UCSB)
  - Design Principles
  - Data Fusion and Fission
  - Elasticity
  - Virtualized Nucleus → Cloud Computing Universe

Internet Chatter



BLOG Wisdom

- “If you want vast, on-demand scalability, you need a non-relational database.” Since scalability requirements:
  - Can change very quickly and,
  - Can grow very rapidly.
- Difficult to manage with a single in-house RDBMS server.
- Although RDBMS scale well:
  - When limited to a single node (scale-up NOT scale-out).
  - Overwhelming complexity to scale on multiple servers.

Application Complexity

```
public void confirm_friend_request(user1, user2)
{
begin_transaction();
      update_friend_list(user1, user2, status.confirmed);
       //user1@Palo Alto Data Center
      update_friend_list(user2, user1, status.confirmed);
      //user2 @London Data Center
end_transaction();
}
```

```
public void confirm_friend_request_A(user1, user2){
  try{          update_friend_list(user1, user2, status.confirmed); //palo alto      }
   catch(exception e){          report_error(e);  return;     }
 try{          update_friend_list(user2, user1, status.confirmed); //london
  }
 catch(exception e) {        revert_friend_list(user1, user2);
      report_error(e);        return;     }
}
```

```
public void confirm_friend_request_B(user1, user2){
try{     update_friend_list(user1, user2, status.confirmed); //palo
alto    }catch(exception
e){     report_error(e);     add_to_retry_queue(operation.updatefriendlis
t, user1, user2);    }
 try{     update_friend_list(user2, user1, status.confirmed); //london
   }catch(exception e)
{    report_error(e);     add_to_retry_queue(operation.updatefriendlist,
user2, user1);    }  }
```

/* get_friends() method has to reconcile results returned by get_friends() because there may be data inconsistency due to a conflict because a change that was applied from the message queue is contradictory to a subsequent change by the user. In this case, status is a bitflag where all conflicts are merged and it is up to app developer to figure out what to do. */

```
public list get_friends(user1){        list actual_friends = new list();        list friends = get_friends();        foreach (friend in friends){            if(friend.status == friendstatus.confirmed){ //no conflict            actual_friends.add(friend);        }else if((friend.status &= friendstatus.confirmed)                and !(friend.status &= friendstatus.deleted)){        // assume friend is confirmed as long as it wasn't also deleted            friend.status = friendstatus.confirmed;                    actual_friends.add(friend);            update_friends_list(user1, friend, status.confirmed);        }else{ //assume deleted if there is a conflict with a delete            update_friends_list( user1, friend, status.deleted)        }    }//foreach   return actual_friends;  }
```

Perspectives

James Hamilton

I love **eventual consistency** but there are some applications that are much easier to implement with strong consistency. Many like eventual consistency because it allows us to scale-out nearly without bound ***but it does come with a cost in programming model complexity.***

February 24, 2010

Outline

- Infrastructure Disruption
  - Enterprise owned => Commodity shared infrastructures
  - Disruptive transformations
- Cloudy Application Landscape
- Clouded Data Management
  - State of the Art lacks “cloud” features
  - Transactional systems
  - Decision support system
- **Gen-next Data Management (UCSB)**
  - **Design Principles**
  - **Data Fusion and Fission**
  - **Elasticity**
  - **Virtualized Nucleus → Cloud Computing Universe**

Design Principles

- **Separate System and Application State**
  - System metadata is critical but small
  - Application data has varying needs
  - Separation allows use of different class of protocols
- **Limit Application interactions to a single node**
  - Allows systems to scale horizontally
  - Graceful degradation during failures
  - Obviate the need for distributed synchronization

Design Principles (contd.)

- **Decouple Ownership from Data Storage**
  - Ownership refers to exclusive read/write access to data
  - Partition ownership – effectively partitions data
  - Decoupling allows light weight ownership transfer
- **Limited distributed synchronization is practical**
  - Maintenance of metadata
  - Provide strong guarantees for data that needs it

Scalability & Elasticity in the Cloud

- **Data Fusion**
  - Enrich Key Value stores [Gstore: ACM SOCC'10, MegaStore: CIDR'11]
- **Data Fission**
  - Cloud enabled relational databases [ElasTraS: HotClouds'09, SQL Azure: ICDE'11, Rcloud: CIDR'11]
- Elasticity of Data Services
- Virtualized Nucleus → Cloud Universe

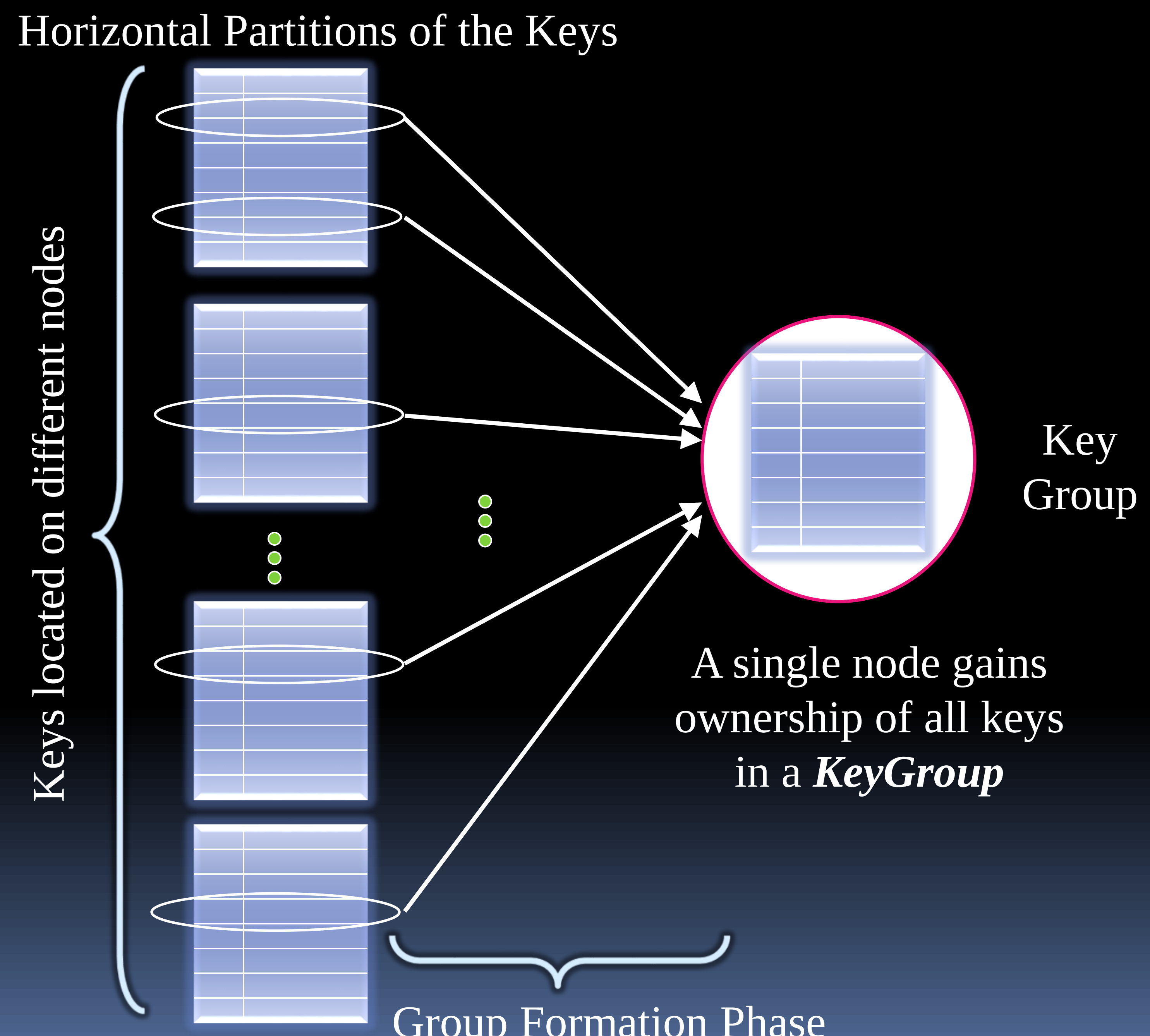
Data Fusion: GStore

Atomic Multi-key Access

- Key value stores:
  - Atomicity guarantees on single keys
  - Suitable for majority of current web applications
- Many other applications warrant multi-key accesses:
  - Online multi-player games
  - Collaborative applications
- Enrich functionality of the Key value stores [Google MegaStore: Static Entity Groups, Transactional Atomicity]

Key Group Abstraction

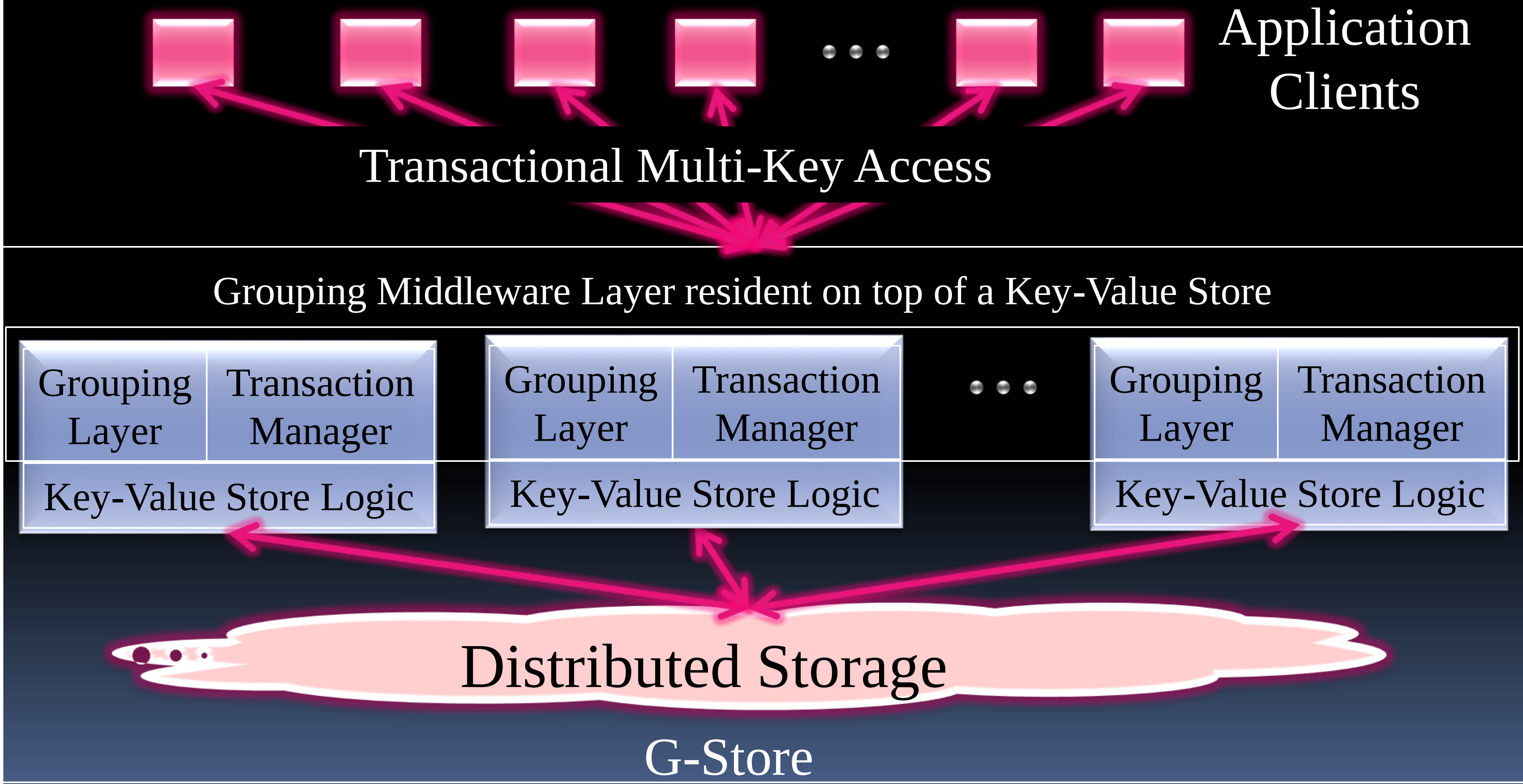
- Define a granule of on-demand transactional access
- Applications select any set of keys
- Data store provides transactional access to the group
- Non-overlapping groups


Horizontal Partitions of the Keys
Keys located on different nodes
Key Group
A single node gains ownership of all keys in a ***KeyGroup***
Group Formation Phase

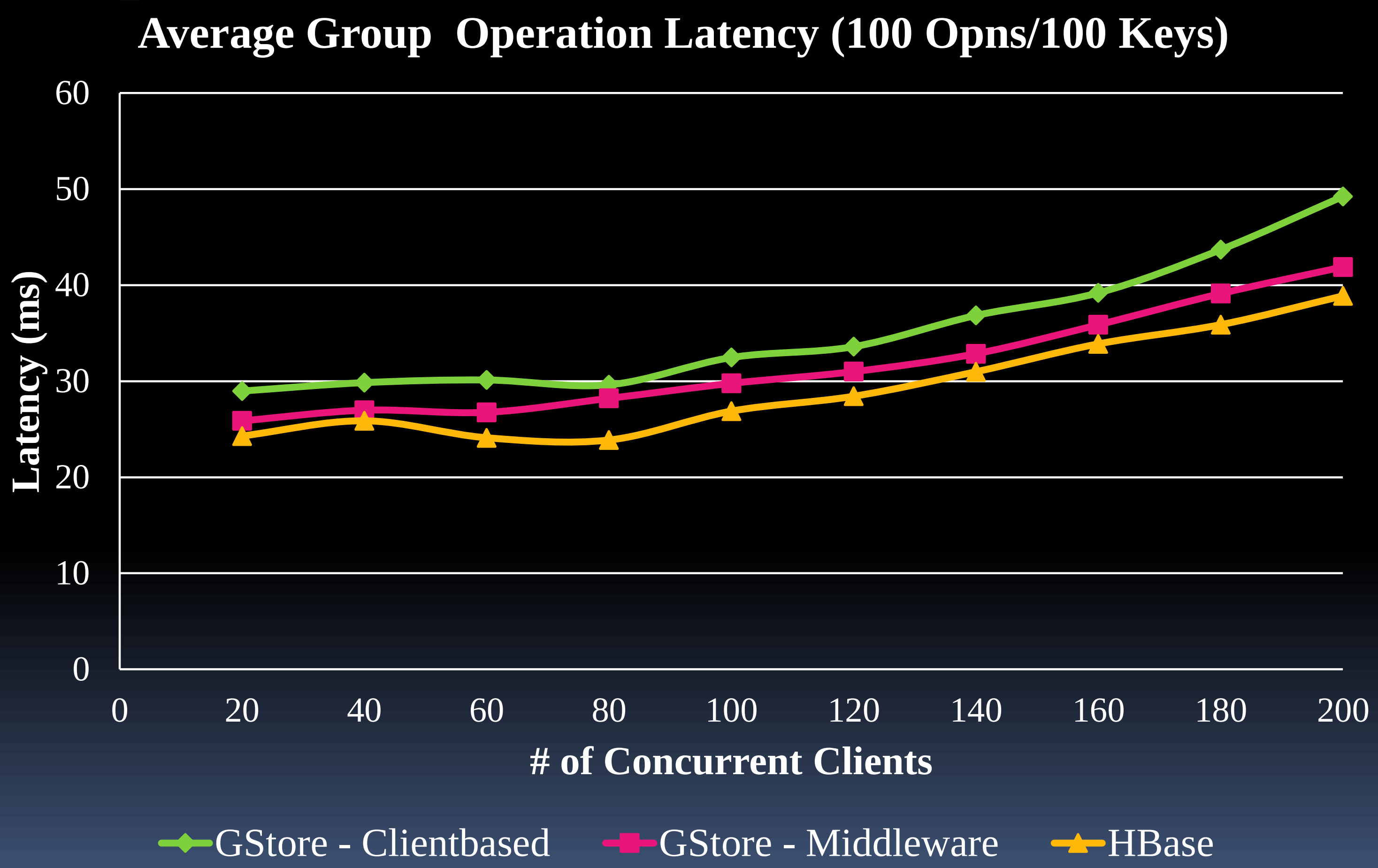
Key Grouping Protocol

- Conceptually akin to “locking”
- Allows collocation of ownership
- Transfer key ownership from “followers” to “leader”
- Guarantee “safe transfer” in the presence of system dynamics:
  - Dynamic migration of data and its control
  - Failures

Implementing GStore



Latency for Group Operations



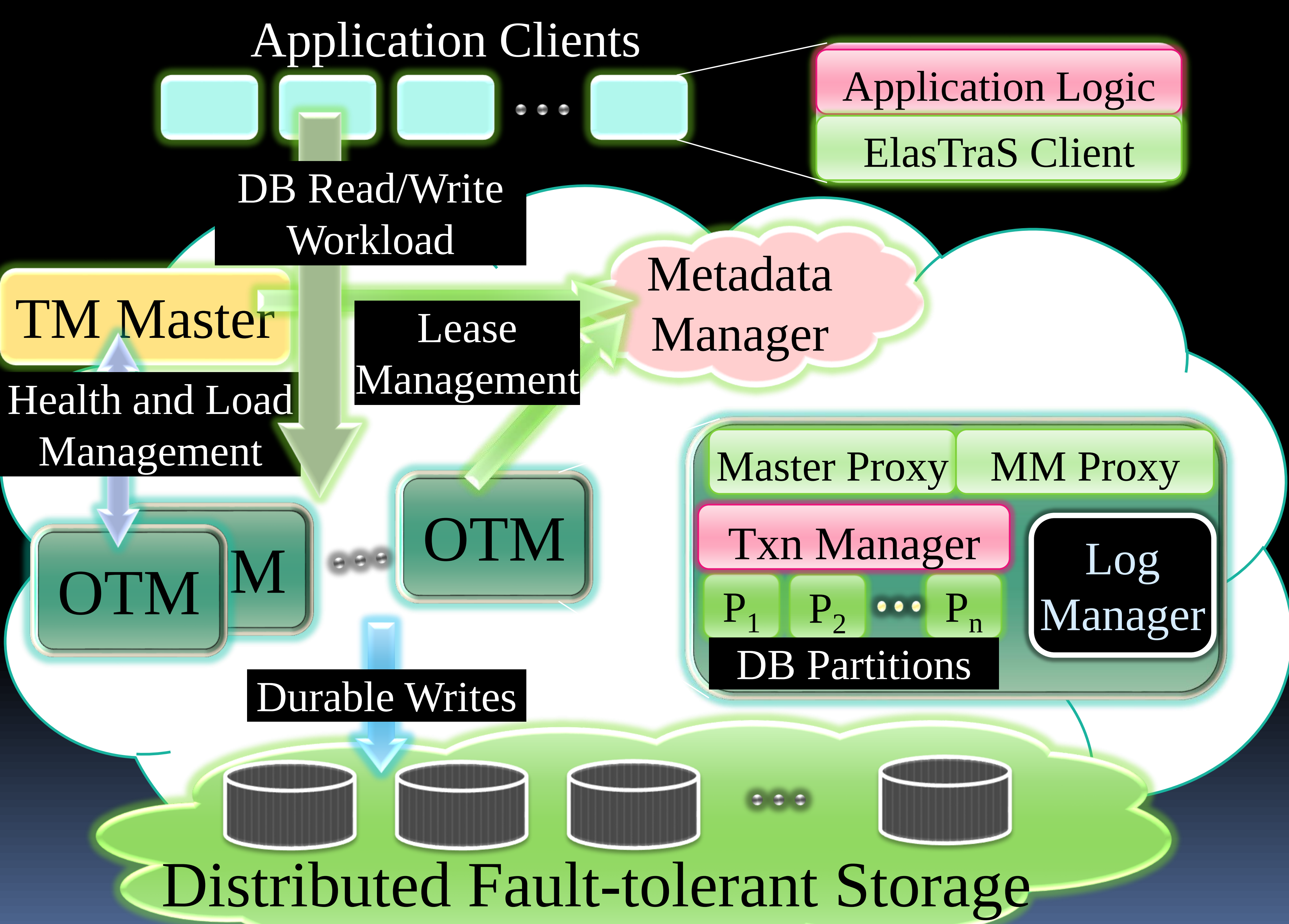
Data Fission: ElasTraS

Elastic Transaction Management

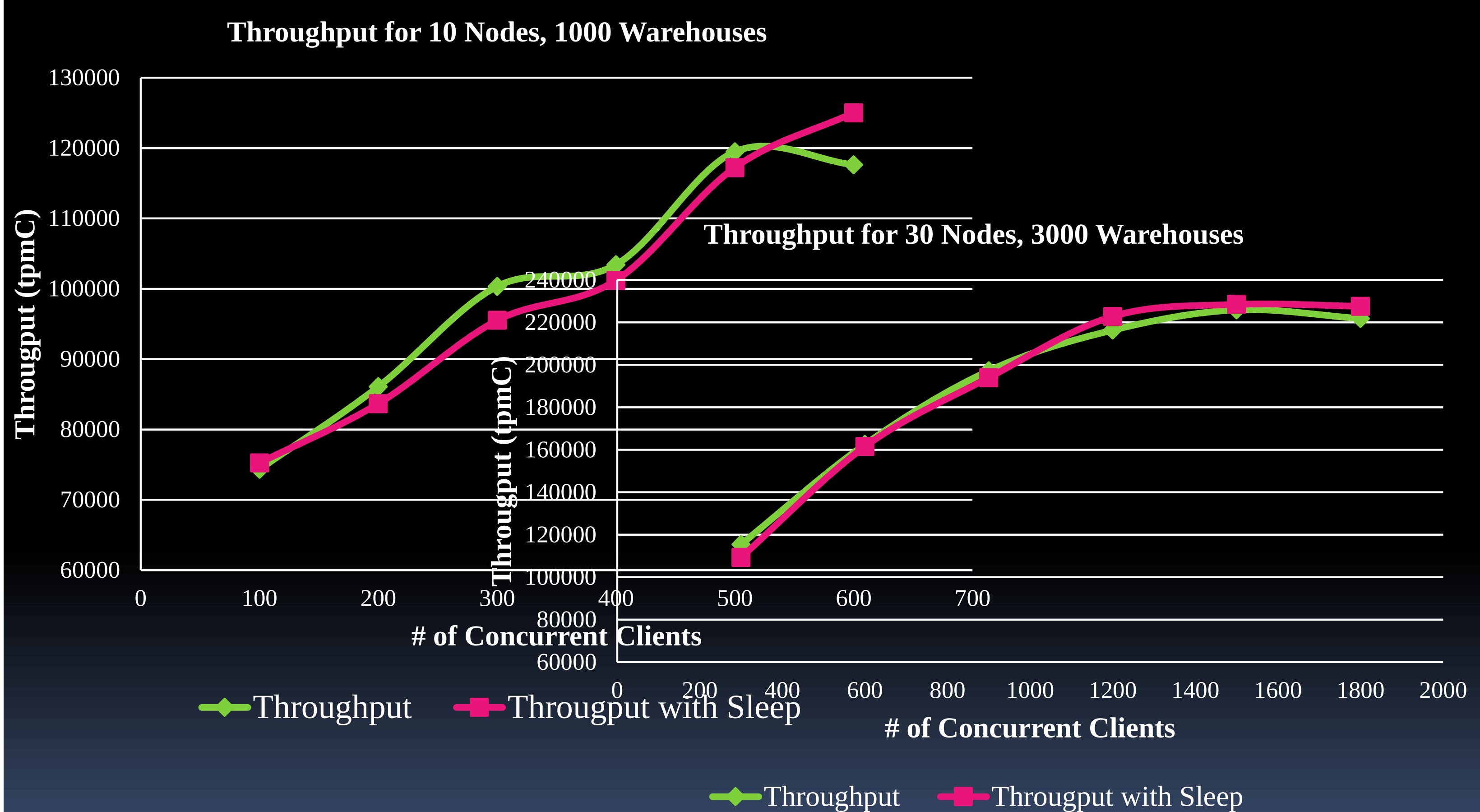
- Designed to make RDBMS cloud-friendly
- Database viewed as a collection of partitions
- Suitable for:
  - Large single tenant database instance
    - Database partitioned at the schema level
  - Multi-tenant database with large number of small databases
    - Each partition is a self contained database

Elastic Transaction Management

- Elastic to deal with workload changes
- Load balance partitions
- Recover from node failures
- Dynamic partition management
- Transactional access to database partitions


Application Clients
Application Logic
ElasTraS Client
DB Read/Write Workload
Metadata Manager
TM Master
Lease Management
Health and Load Management
OTM
M
OTM
Master Proxy
MM Proxy
Txn Manager
Log Manager
P_1 P_2 P_n
DB Partitions
Durable Writes
Distributed Fault-tolerant Storage

Throughput



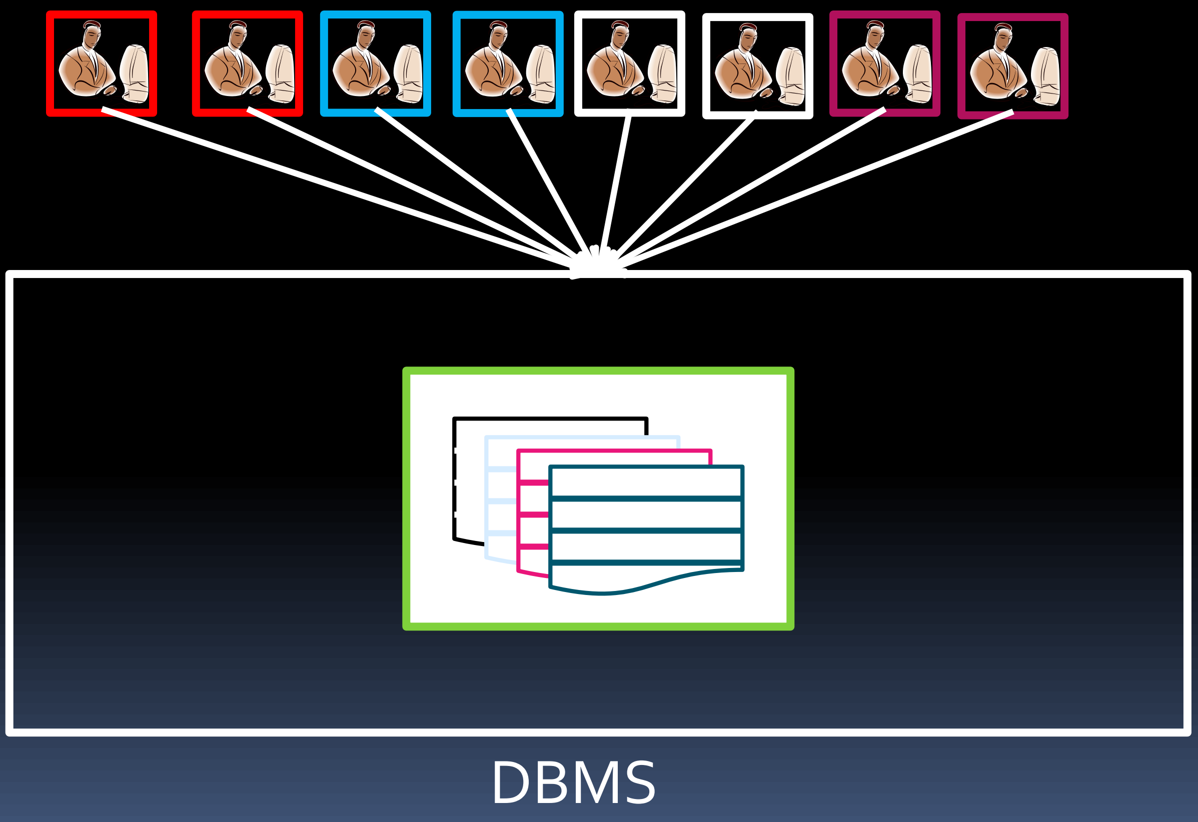
Throughput for 10 Nodes, 1000 Warehouses
130000
120000
110000
100000
90000
80000
70000
60000
Througput (tpmC)
0
100
200
300
400
500
600
700
# of Concurrent Clients
Throughput
Througput with Sleep
Throughput for 30 Nodes, 3000 Warehouses
240000
220000
200000
180000
160000
140000
120000
100000
80000
60000
Througput (tpmC)
0
200
400
600
800
1000
1200
1400
1600
1800
2000
# of Concurrent Clients
Throughput
Througput with Sleep

Elasticity in the Cloud: Live Data Migration

Elasticity

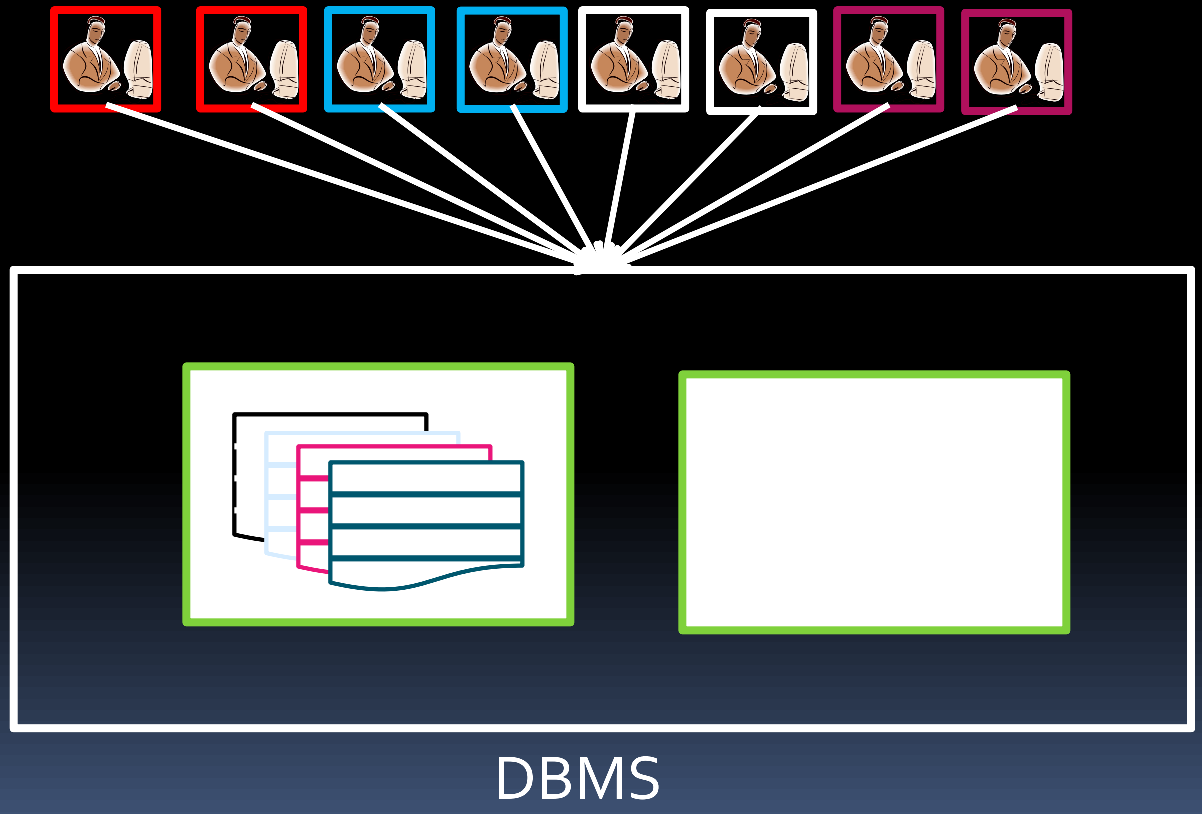
- A database system built over a pay-per-use infrastructure
  - Infrastructure as a Service for instance
- Scale up and down system size on demand
  - Utilize peaks and troughs in load
- Minimize operating cost while ensuring good performance

Elasticity in the Database Layer



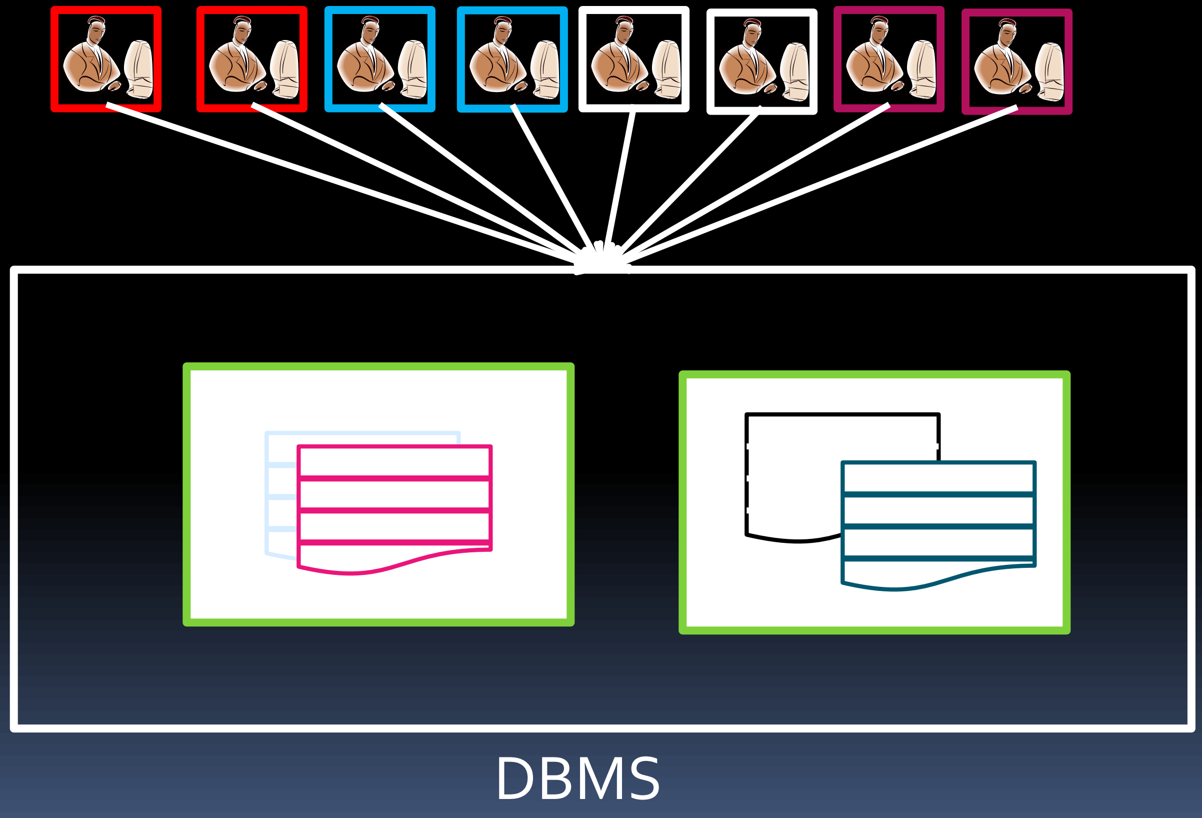
Elasticity in the Database Layer

Capacity expansion to deal with high load – Guarantee good performance



Elasticity in the Database Layer

Consolidation during periods of low load – Cost Minimization



Live Database Migration

- All Elasticity induced dynamics in a Live system
- Minimal service interruption for migrating data fragments
  - Minimize operations failing
  - Minimize unavailability window, if any
- Negligible performance impact
- No overhead during normal operation
- Guaranteed safety and correctness

Live Database Migration
Current State – A teaser

- **Shared storage** architecture
  - **Proactive** state migration
    - No need to migrate persistent data
    - Migrate database cache and transaction state proactively
    - Ensures low performance impact
- **Shared nothing** architecture
  - **Reactive** state migration
    - Migrate minimal database state
    - Persistent image migrated asynchronously on demand
- More details to follow in the near future
  - A long presentation in its own merit

Virtualized Nucleus to Cloud Computing Universe: Current Work

Cloud Abstractions

BigTable Semantics
Single-key ATOMIC Read
Single-key ATOMIC Write
Single-key ATOMIC Read-modify-Write

GFS SEMANTICS	WRITE	READ
SERIAL	DEFINED	DEFINED but may be INCONSISTENT
CONCURRENT	CONSISTENT but UNDEFINED	DEFINED but may be INCONSISTENT
FAILURE	INCONSISTENT	INCONSISTENT

Cloud Abstractions

Higher Level Abstractions: Multi-key Atomicity while maintaining Scalability, Elasticity, Fault-tolerance, & Self-Manageability

BigTable Semantics
Single-key ATOMIC Read
Single-key ATOMIC Write
Single-key ATOMIC Read-modify-Write

Concluding Remarks

- Data Management for Cloud Computing poses a fundamental challenges:
  - Scalability, Reliability, **Elasticity, Payment Model,** Data Consistency
- Cloud Computing in other sectors:
  - Information Technology in Government, Health-care etc.
  - Scientific Computing and Large-scale Science Data
- Finally, the computing substrate will also evolve:
  - Multiple Data Centers
  - Leveraging the Network Edge (beyond content caching)
- Security and Privacy of Data and Infrastructure in the Cloud

Cloud Computing at UCSB & Santa Barbara

Research Activities

- Cloud Computing Infrastructures:
  - Rich Wolski, UCSB
- Cloud Programming Models, Applications and Languages:
  - Chadra Krintz, UCSB
- Data Management in Clouds:
  - Divy Agrawal & Amr El Abbadi, UCSB
- Security & Privacy Models in Clouds:
  - Giovanni Vigna & Christopher Kruegel, UCSB

Industrial Start-ups

- Cloud Computing Infrastructures:
  - Eucalyptus: Rich Wolski
- Cloud Computing Management:
  - RightScale: Thurston von Eicken
- Application Hosting in the Cloud:
  - AppFolio: Klaus Schauser